[Elasticsearch: 权威指南](https://elasticsearch.cn/book/elasticsearch_definitive_guide_2.x/%20)

1. **ES概念**



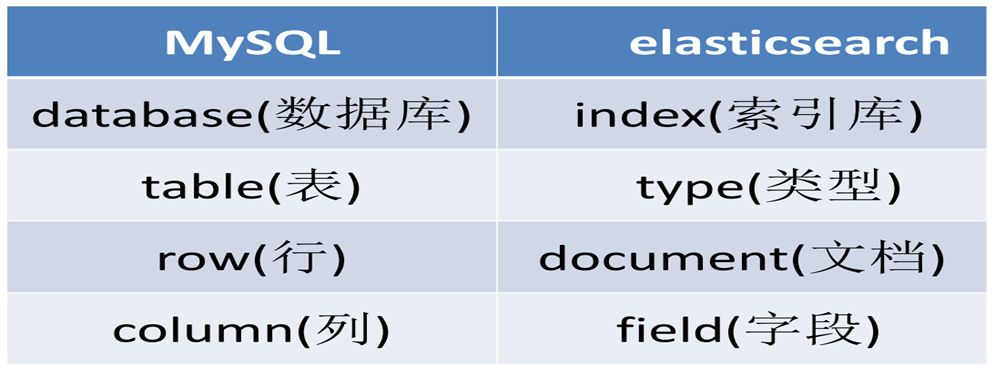
http://cwiki.apachecn.org/pages/viewpage.action?pageId=10028661

ES是一个最终一致性系统

ES是密集使用磁盘的应用

cluster（集群），node(节点)，index（索引），type(类型)，document（文档），field（字段）,mapping(定义了index中各个字段的类型和属性等，相当于MySQL的table定义语句)

MySQL 与 elasticsearch 的概念对应关系：



（注意，ES中一个index下最好只创建一个type，不要创建多个type！）

### 如何确定分片数

* 1. ES是静态分片，一旦分片数在创建索引时确定那么后继不能修改
  2. 数据量在亿级别，8或者16分片够用，**分片数最好是2的n次方**。
  3. 如果后继数据量的增长超过创建索引的预期，那么需要创建新索引并重灌数据
  4. ES维护团队后继会提供重灌数据的服务
  5. 创建mapping是请自行制定分片数，否则创建的索引的分片数是ES的默认值。这其实并不符合需求。
  6. 副本数：一般设置为1

## 存储与搜索

ES是Elastic Search的缩写，ES是基于Lucene的分布式存储；Lucene提供了全文检索的功能，ES在此之上加入索引分布式的机制，提供了数据分片、数据副本、数据同步等功能，保证了数据的安全性；

ES存储的基本单位是一个Document，可以想象是数据库中的一行，与数据库类似的是，一个Document包含有多个Field，可以是数值类型也可以是字符类型，与数据库不同的是，Document可以随意添加或删除字段，也就是Schema less，另外ES对于需要检索的字段可以声明为需要索引，这样可以很方便就对Document进行检索。

ES具有以下特性：

1.分布式的实时文件存储，每个字段都被索引并可被搜索；

2.分布式的实时分析搜索引擎；

3.可以扩展到上百台服务器，处理PB级结构化或非结构化数据；

ES在分布式概念上做了很大程度上的透明化，你不需要知道任何关于分布式系统、分片、集群发现或者其他大量的分布式概念。ES致力于隐藏分布式系统的复杂性，以下这些操作都是在底层自动完成的：

1.将你的文档分区到不同的容器或者分片(shards)中，它们可以存在于一个或多个节点中。

2.将分片均匀的分配到各个节点，对索引和搜索做负载均衡。

3.冗余每一个分片，防止硬件故障造成的数据丢失。

4.将集群中任意一个节点上的请求路由到相应数据所在的节点。

5.无论是增加节点，还是移除节点，分片都可以做到无缝的扩展和迁移。

## 性能

基于以上存储特性，ES可以处理PB级的数据，在我们的压测过程中，ES的表现也非常出色；

在五十亿条数据基量（每条数据大小800B）情况下，多线程写入测试，测试结果TPS可以稳定到 8W/S 6W/S，

在同样数据基量情况下，基于document\_id进行多线程搜索排序测试，测试结果为8W+/S ；

[性能优化措施](http://jpcloud.jd.com/pages/viewpage.action?pageId=15337675)

## 容灾

ES集群中，数据被分散（hash）到多个分片中，每个分片根据设置会有一到二个冗余的分片副本，主分片以及副本分片被分散到不同的节点上；

这保证了在硬件故障时数据不会丢失；

此外，集群中的任意一个节点都会将请求路由到相应数据所在的节点，这样访问集群中的任意节点都会得到正常的数据返回

1. **如何提高写性能**

[Performance Considerations for Elasticsearch Indexing](https://www.elastic.co/blog/performance-considerations-elasticsearch-indexing)

1. **如何提高读性能**

<https://www.elastic.co/blog/found-optimizing-elasticsearch-searches>

# ES性能调优

## 减少refresh次数

LUCENE在新增数据时，为了提高写的性能，采用的是延迟写入的策略，即将数据先写入到内存，当延时超过1秒（默认）时，会出发一次refresh，refresh回把要写入的数据以段的形式刷新到操作系统的文件缓存系统中。当数据以段的形式刷新到文件缓存系统后才可以搜索，所以在搜索的时效性要求不高时，可以增加延时时间，这样可以减少段的数量，为后面段合并减少压力。但是也会消耗更多的Heap内存和搜索延时。

默认的刷新时间间隔是1s，对于写入量很大的场景，这样的配置会导致写入吞吐量很低，适当提高刷新间隔，可以提升写入量，代价就是让新写入的数据在60s之后可以被搜索，新数据可见的及时性有所下降。

"index.refresh\_interval": "60s"

## 减少Flush的次数

当Tranlog的数据量达到512MB或者30分钟（默认），会出发一次刷新（Flush）。刷新主要是将文件缓存系统中的段持久化到磁盘中，因为持久化到磁盘是一个比较耗时的操作。可以通过配置增加Tranlog的缓存数据量。

Index.Tranlog.flush\_threshold\_size=1GB

这样配置意味着文件缓存系统要存储更多的数据，所以要减少Heap的内存空间，为操作系统的文件缓存系统留下更多的空间

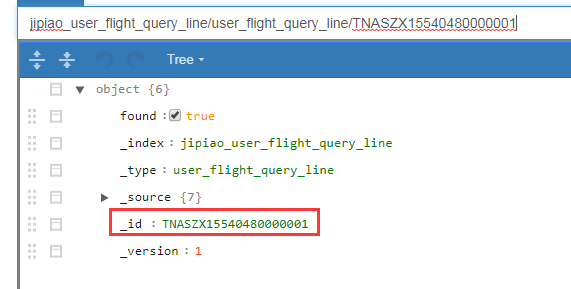
# Java中Es操作

## 查询

jipiao\_user\_flight\_query\_line/user\_flight\_query\_line/\_search?q=queryDate:2019-04-01

### Id查询

jipiao\_user\_flight\_query\_line/user\_flight\_query\_line/TNASZX15540480000001



client.prepareGet(*INDEX\_NAME*, *TYPE\_NAME*, docId)  
 .execute()  
 .actionGet();

### [QueryBuilder简单查询](https://blog.csdn.net/lom9357bye/article/details/52852533)

**matchAllQuery()**

matchQuery("filedname","value")匹配单个字段，匹配字段名为filedname,值为value的文档

**multiMatchQuery(Object text, String... fieldNames)**

**wildcardQuery(String name，String query)模糊查询，name中包含query**

### 指定前缀查询

QueryBuilders.prefixQuery("spell", entity.getSpell())

### TermQuery

精确匹配



[elasticsearch 查询（match和term）](https://www.cnblogs.com/yjf512/p/4897294.html)

## bool查询的使用

Must代表and，should代表or

Bool查询对应Lucene中的BooleanQuery，它由一个或者多个子句组成，每个子句都有特定的类型。

#### must

返回的文档必须满足must子句的条件，并且参与计算分值

#### filter

返回的文档必须满足filter子句的条件。但是不会像Must一样，参与计算分值

#### should

返回的文档可能满足should子句的条件。在一个Bool查询中，如果没有must或者filter，有一个或者多个should子句，那么只要满足一个就可以返回。minimum\_should\_match参数定义了至少满足几个子句。

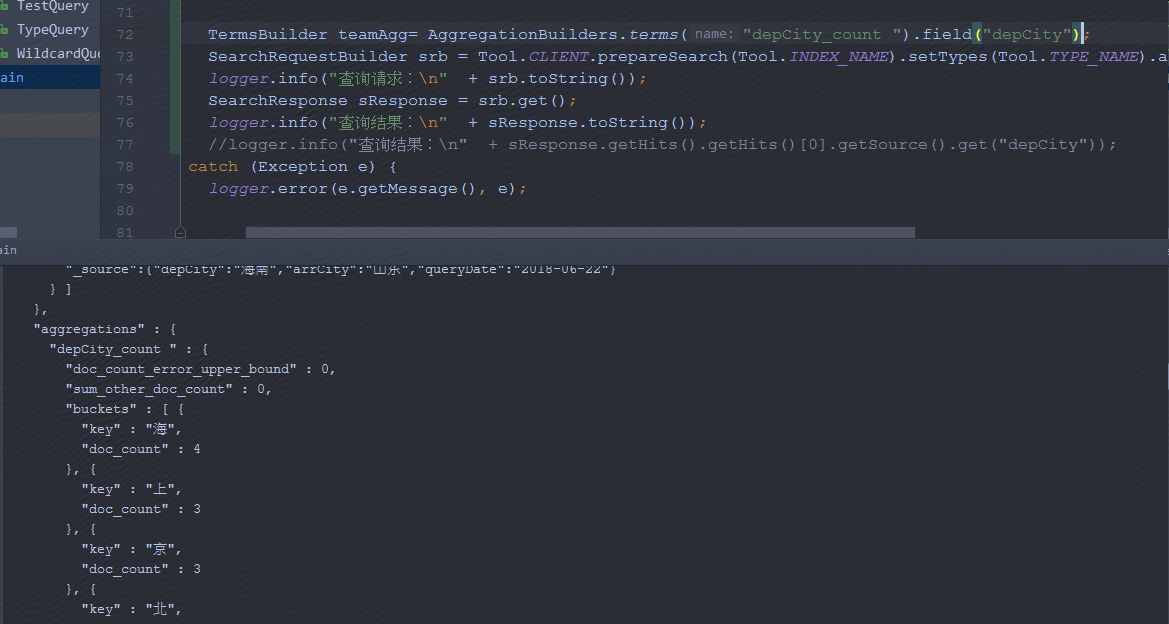
#### must\_nout

返回的文档必须不满足must\_not定义的条件。

### 聚合查询Aggregation

[ElasticSearch AggregationBuilders java api常用聚会查询](https://www.cnblogs.com/xionggeclub/p/7975982.html)

[ElasticSearch 的 聚合（Aggregations）](https://blog.csdn.net/mlljava1111/article/details/70598519)



"aggregations" : {

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "海",

"doc\_count" : 4

}, {

"key" : "上",

"doc\_count" : 3

}, {

"key" : "京",

"doc\_count" : 3

}, {

"key" : "北",

"doc\_count" : 2

}, {

"key" : "南",

"doc\_count" : 2

}, {

"key" : "安",

"doc\_count" : 1

}, {

"key" : "西",

"doc\_count" : 1

} ]

}

}

## 更新（insert、update、delete）

对一个文档进行覆盖，修改，删除，它的版本值都会增加。好好体会一下。

新文档的版本从1开始递增。如果删除一个文档后，再重新插入相同id的文档，那么它的版本并不是从1开始。注意。看下面的例子。

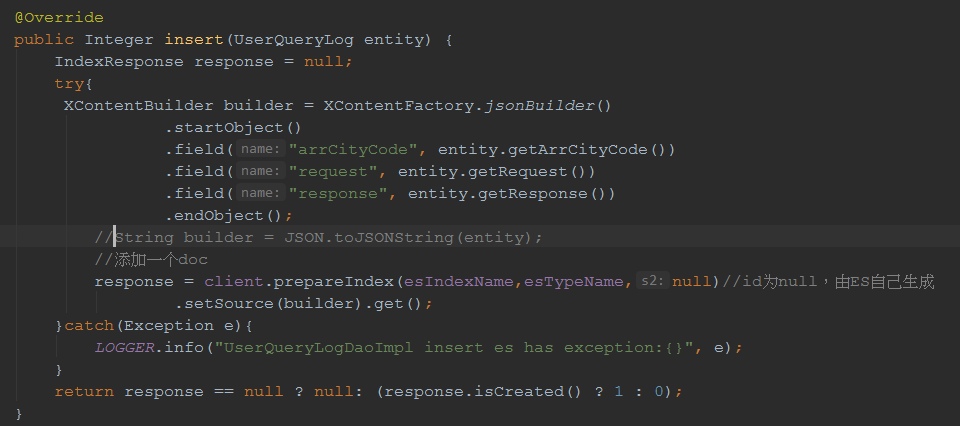
Update：

Tool.*CLIENT*.prepareUpdate(*IndexName*,*TypeName*,"1").setDoc("{\"age\":2}").get();

Insert：

IndexResponse response = Tool.*CLIENT*.prepareIndex(indexName, typeName, docId)  
 .setSource(json)  
 .execute()  
 .actionGet();

除了json串，还可以使用



Delete

DeleteResponse dResponse = Tool.*CLIENT*.prepareDelete(*IndexName*,*TypeName*,"1").get();

## 分词器：

[Elasticsearch 中文分词器 IK 配置和使用](https://blog.csdn.net/jam00/article/details/52983056)

not\_analyzed不进行分析，整个词作为keyWords

Index属性：Analyzed VS not\_analyzed VS No

Analyzed是使用分词（但是长度超过31766会非常消耗机器资源），一段描述性的文字：“原产地广东佛山”就会被分词为“原产地”、“广东”、“佛山”，这样用户在查询“佛山”时就能找到这段话；如果不分词就找不到

Not\_analyzed是不使用分词，一个电话号码：“010-62256225”不分词，那么用“62256225”查询不到，只能用“010-62256225”查询

No是不索引，这样的内容不能被查询到

最后，要知道的一点：不管上述analyzer还是not\_analyzer, 经过这样的处理之后，这玩意就开始有个学名了：token

如果设置字段的index属性为not\_analyzed，原始文本将作为单个分词，其最大长度跟UTF8 编码有关，默认的最大长度是32766Bytes，如果字段的文本超过该限制，那么ElasticSearch将跳过（Skip）该文档，并在Response中抛出异常消息：

operation[607]: index returned 400 \_index: ebrite \_type: events \_id: 76860 \_version: 0 error: Type: illegal\_argument\_exception Reason: "Document contains at least one immense term in field="event\_raw" (whose UTF8 encoding is longer than the max length 32766), all of which were skipped. Please correct the analyzer to not produce such terms. The prefix of the first immense term is: '[112, 114,... 115]...', original message: bytes can be at most 32766 in length; got 35100" CausedBy:Type: max\_bytes\_length\_exceeded\_exception Reason: "bytes can be at most 32766 in length; got 35100"

可以在字段中设置ignore\_above属性，该属性值指的是字符数量，而不是字节数量；由于一个UTF8字符最多占用3个字节，因此，可以设置

code

1“ignore\_above”：10000

这样，超过30000字节之后的字符将会被分析器忽略，单个分词（Term）的最大长度是30000Bytes。

The value forignore\_aboveis thecharacter count, but Lucene counts bytes. If you use UTF-8 text with many non-ASCII characters, you may want to set the limit to32766 / 3 = 10922since UTF-8 characters may occupy at most 3 bytes.

## store

[ES系列十一、ES的index、store、\_source、copy\_to和all的区别](https://www.cnblogs.com/wangzhuxing/p/9527151.html)

## ignore\_above

在index配置为index:not\_ analyzed时候，会限制字符不能超过32766

原文链接 : <https://www.elastic.co/guide/en/elasticsearch/reference/5.3/ignore-above.html>

译文链接 : <http://www.apache.wiki/pages/createpage-entervariables.action?templateId=4816898&spaceKey=Elasticsearch&title=&newSpaceKey=Elasticsearch&fromPageId=9405287>

字符串长度超过 **ignore\_above** 设置的不会被索引和存储.

|  |  |  |
| --- | --- | --- |
| curl -XPUT 'localhost:9200/my\_index?pretty' -H 'Content-Type: application/json' -d'  {    "mappings": {      "my\_type": {        "properties": {          "message": {            "type": "keyword",            "ignore\_above": 20  # 1          }        }      }    }  }  '  curl -XPUT 'localhost:9200/my\_index/my\_type/1?pretty' -H 'Content-Type:  application/json' -d' # 2  {    "message": "Syntax error"  }  '  curl -XPUT 'localhost:9200/my\_index/my\_type/2?pretty' -H 'Content-Type:  application/json' -d' # 3  {    "message": "Syntax error with some long stacktrace"  }  '  curl -XGET 'localhost:9200/\_search?pretty' -H 'Content-Type: application/json' -d' # 4  {    "aggs": {      "messages": {        "terms": {          "field": "message"        }      }    }  } | | |
| 1 | 字段会忽略超过20个字符的字符串. |
| 2 | 这个 **document** 成功被索引. |
| 3 | 这个 **document** 会被索引,但是不会索引 **message** 字段. |
| 4 | 搜索返回两个文档,但只有第一个存在于 **terms**（词条）聚合中. |

**tip**

**ignore\_above**设置允许在相同索引的相同名称的字段有不同的配置,可以使用 [PUT mapping API](https://www.elastic.co/guide/en/elasticsearch/reference/5.3/indices-put-mapping.html) 在现有字段上更新其值。

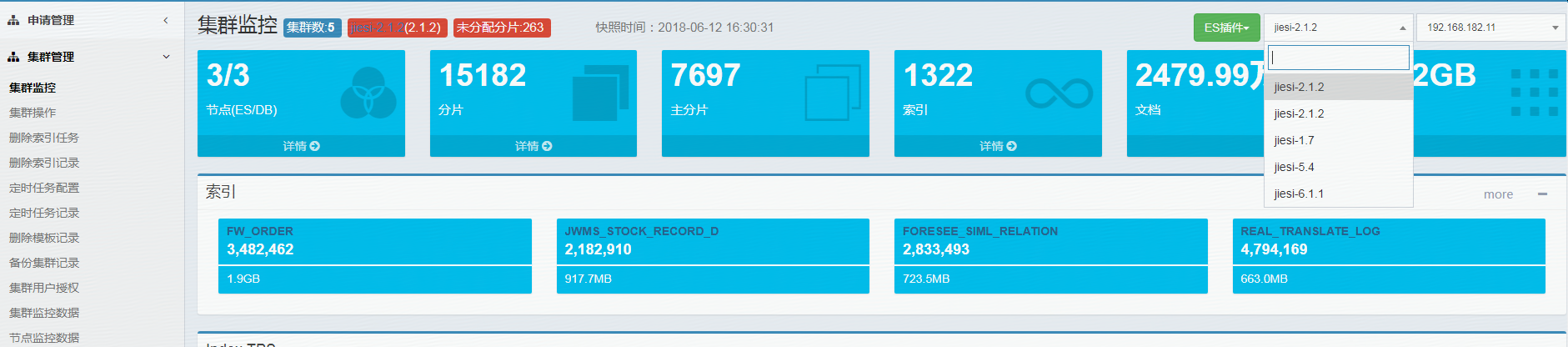
此选项对于防止Lucene term（词条）长度超过32766是有用的.

**ignore\_above**的值是字符数，但 **Lucene**计数的是字节。所以如果你使用具有许多非ASCII字符的UTF-8文本，则可能需要将限制设置为 **32766/3 = 10922**，因为UTF-8字符可能占用至多3个字节。

# 实战

测试环境：test.esm.jd.com

集群：jiesi-2.1.2



节点：点击详情



## 新建mapping



{

"settings": {

"index": {

"number\_of\_shards": "4",

"number\_of\_replicas": "1"

}

},

"mappings": {

"air\_flight\_query\_type": {

"\_ttl": {

"enabled": false

},

"dynamic": false,

"\_all": {

"enabled": false

},

"properties": {

"depCity": {

"type": "string",

"index": "not\_analyzed"

},

"arrCity": {

"type": "string",

"index": "not\_analyzed"

},

"queryDate": {

"type": "date",

"format": "yyyy-MM-dd HH:mm:ss",

"index": "not\_analyzed"

},

"userPin": {

"type": "string",

"index": "not\_analyzed"

}

}

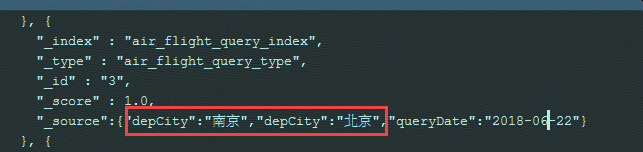
}

}

}

## 数据准备（插入文档）

public static void prepareTestDoc(){  
  
 String json1 = "{" +  
 "\"depCity\":\"北京\"," +  
 "\"arrCity\":\"上海\"," +  
 "\"queryDate\":\"2018-06-22 00:00:00\"," +  
 "\"userPin\":\"huyanxia\"" +  
 "}";  
  
 String json2 = "{" +  
 "\"depCity\":\"北京\"," +  
 "\"arrCity\":\"三亚\"," +  
 "\"queryDate\":\"2018-06-21 00:00:00\"," +  
 "\"userPin\":\"huyanxia1\"" +  
 "}";  
  
 String json3 = "{" +  
 "\"depCity\":\"南京\"," +  
 "\"arrCity\":\"北京\"," +  
 "\"queryDate\":\"2018-06-25 00:00:00\"," +  
 "\"userPin\":\"huyanxia2\"" +  
 "}";  
  
 String json4 = "{" +  
 "\"depCity\":\"西安\"," +  
 "\"arrCity\":\"延安\"," +  
 "\"queryDate\":\"2018-06-22 00:00:00\"," +  
 "\"userPin\":\"huyanxia\"" +  
 "}";  
  
 String json5 = "{" +  
 "\"depCity\":\"海南\"," +  
 "\"arrCity\":\"山东\"," +  
 "\"queryDate\":\"2018-06-22 00:00:00\"," +  
 "\"userPin\":\"huyanxia\"" +  
 "}";  
  
 String json6 = "{" +  
 "\"depCity\":\"上海\"," +  
 "\"arrCity\":\"北京\"," +  
 "\"queryDate\":\"2018-06-19 00:00:00\"," +  
 "\"userPin\":\"huyanxia\"" +  
 "}";  
  
 String json7 = "{" +  
 "\"depCity\":\"上海\"," +  
 "\"arrCity\":\"北京\"," +  
 "\"queryDate\":\"2018-06-22 00:00:00\"," +  
 "\"userPin\":\"huyanxia\"" +  
 "}";  
  
 String json8 = "{" +  
 "\"depCity\":\"上海\"," +  
 "\"queryDate\":\"2018-06-22 00:00:00\"," +  
 "\"userPin\":\"huyanxia5\"" +  
 "}";  
  
  
 *indexWithStr*(Tool.*INDEX\_NAME*, Tool.*TYPE\_NAME*,"1",json1);  
 *indexWithStr*(Tool.*INDEX\_NAME*, Tool.*TYPE\_NAME*,"2",json2);  
 *indexWithStr*(Tool.*INDEX\_NAME*, Tool.*TYPE\_NAME*,"3",json3);  
 *indexWithStr*(Tool.*INDEX\_NAME*, Tool.*TYPE\_NAME*,"4",json4);  
 *indexWithStr*(Tool.*INDEX\_NAME*, Tool.*TYPE\_NAME*,"5",json5);  
 *indexWithStr*(Tool.*INDEX\_NAME*, Tool.*TYPE\_NAME*,"6",json6);  
 *indexWithStr*(Tool.*INDEX\_NAME*, Tool.*TYPE\_NAME*,"7",json7);  
 *indexWithStr*(Tool.*INDEX\_NAME*, Tool.*TYPE\_NAME*,"8",json8);  
}



## 查询

### 全部查询

QueryBuilder qb = QueryBuilders.matchAllQuery();

### 根据docId查询

GetResponse response = Tool.*CLIENT*.prepareGet(Tool.*INDEX\_NAME*,Tool.*TYPE\_NAME*,"1").get();  
*logger*.info("查询结果：\n" + response.getSourceAsMap());

查询结果：

{userPin=huyanxia, queryDate=2018-06-22 00:00:00, arrCity=上海, depCity=北京}

### 精确复合查询

QueryBuilder queryBuilder1 = QueryBuilders.matchQuery("depCity", "上海");  
 QueryBuilder queryBuilder2 = QueryBuilders.matchQuery("arrCity", "北京");  
  
 BoolQueryBuilder boolQueryBuilder = QueryBuilders.boolQuery();  
 boolQueryBuilder.must(queryBuilder1);  
 boolQueryBuilder.must(queryBuilder2);

### date日期范围

Sql:

Select \* from table where queryDate>="2018-06-21 00:00:00" and queryDate<="2018-06-24 00:00:00"

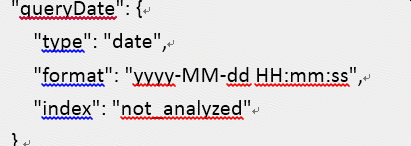
Java api:

BoolQueryBuilder boolQueryBuilder = QueryBuilders.*boolQuery*();  
boolQueryBuilder.must(QueryBuilders.*rangeQuery*("queryDate").from("2018-06-21 00:00:00").to("2018-06-24 00:00:00"));

SearchRequestBuilder srb = Tool.*CLIENT*.prepareSearch(Tool.*INDEX\_NAME*).setTypes(Tool.*TYPE\_NAME*).setQuery(boolQueryBuilder).setSize(20);//setSize(3) 可以设置查询的文档数的大小，默认10

下面也可以，可以控制是开还是闭空间、

Mapping中存储的是



查询使用年月日，默认后面时分秒是00:00:00

Select \* from table where queryDate>"2018-06-21 00:00:00" and queryDate<"2018-06-24 00:00:00"

boolQueryBuilder.must(QueryBuilders.*rangeQuery*("queryDate").format("yyyy-MM-dd").gt("2018-06-21").lt("2018-06-24"));

### 一个Group by 和一个order by

Sql:

Select count(\*) as depCity\_count from table where queryDate>="2018-06-21 00:00:00" and queryDate<="2018-06-24 00:00:00" group by depCity order by depCity\_count asce

Java Api:

BoolQueryBuilder boolQueryBuilder = QueryBuilders.*boolQuery*();  
boolQueryBuilder.must(QueryBuilders.*rangeQuery*("queryDate").from("2018-06-21 00:00:00").to("2018-06-24 00:00:00"));

TermsBuilder teamAgg= AggregationBuilders.*terms*("depCity\_count ").field("depCity").subAggregation(AggregationBuilders.*count*("depCity\_count").field("depCity")).order(Terms.Order.*aggregation*("depCity\_count", true));

SearchRequestBuilder srb = Tool.*CLIENT*.prepareSearch(Tool.*INDEX\_NAME*).setTypes(Tool.*TYPE\_NAME*).setQuery(boolQueryBuilder).addAggregation(teamAgg).setSize(20);//setSize(3) 可以设置查询的文档数的大小，默认10  
*logger*.info("查询请求：\n" + srb.toString());  
SearchResponse sResponse = srb.get();  
*logger*.info("查询结果：\n" + sResponse.toString());

结果：先是查询结果然后再聚合

{

"took" : 1,

"timed\_out" : false,

"\_shards" : {

"total" : 4,

"successful" : 4,

"failed" : 0

},

"hits" : {

"total" : 6,

"max\_score" : 1.0,

"hits" : [ {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "7",

"\_score" : 1.0,

"\_source":{"depCity":"上海","arrCity":"北京","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "8",

"\_score" : 1.0,

"\_source":{"depCity":"上海","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia5"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "1",

"\_score" : 1.0,

"\_source":{"depCity":"北京","arrCity":"上海","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "2",

"\_score" : 1.0,

"\_source":{"depCity":"北京","arrCity":"三亚","queryDate":"2018-06-21 00:00:00","userPin":"huyanxia1"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "4",

"\_score" : 1.0,

"\_source":{"depCity":"西安","arrCity":"延安","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "5",

"\_score" : 1.0,

"\_source":{"depCity":"海南","arrCity":"山东","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

} ]

},

"aggregations" : {

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "海南",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

}, {

"key" : "西安",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

}, {

"key" : "上海",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

}, {

"key" : "北京",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

} ]

}

}

}

### 多个group，多个order

Sql:

Select count(\*) as depCity\_count from table where queryDate>="2018-06-21 00:00:00" and queryDate<="2018-06-24 00:00:00" group by queryDate,depCity order by queryDate asc,depCity\_count desc

BoolQueryBuilder boolQueryBuilder = QueryBuilders.*boolQuery*();  
 boolQueryBuilder.must(QueryBuilders.*rangeQuery*("queryDate").from("2018-06-21 00:00:00").to("2018-06-24 00:00:00"));  
 TermsBuilder teamAgg= AggregationBuilders.*terms*("depCity\_count ").field("depCity").subAggregation(AggregationBuilders.*count*("depCity\_count").field("depCity")).order(Terms.Order.*aggregation*("depCity\_count", true));  
 TermsBuilder dateAgg= AggregationBuilders.*terms*("queryDate\_count").field("queryDate").order(Terms.Order.*term*(true)).subAggregation(teamAgg);

SearchRequestBuilder srb = Tool.*CLIENT*.prepareSearch(Tool.*INDEX\_NAME*).setTypes(Tool.*TYPE\_NAME*).setQuery(boolQueryBuilder).addAggregation(dateAgg).setSize(20);//setSize(3) 可以设置查询的文档数的大小，默认10  
  
 *logger*.info("查询请求：\n" + srb.toString());  
 SearchResponse sResponse = srb.get();  
 *logger*.info("查询结果：\n" + sResponse.toString());

**数据一的请求与结果：**

**请求：**

**{**

**"size" : 20,**

**"query" : {**

**"bool" : {**

**"must" : {**

**"range" : {**

**"queryDate" : {**

**"from" : "2018-06-21",**

**"to" : "2018-06-24",**

**"format" : "yyyy-MM-dd",**

**"include\_lower" : true,**

**"include\_upper" : true**

**}**

**}**

**}**

**}**

**},**

**"aggregations" : {**

**"queryDate\_count" : {**

**"terms" : {**

**"field" : "queryDate",**

**"order" : {**

**"\_term" : "desc"**

**}**

**},**

**"aggregations" : {**

**"depCity\_count " : {**

**"terms" : {**

**"field" : "depCity",**

**"order" : {**

**"depCity\_count" : "desc"**

**}**

**},**

**"aggregations" : {**

**"depCity\_count" : {**

**"value\_count" : {**

**"field" : "depCity"**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

**}**

结果：

"aggregations" : {

"queryDate\_count" : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : 1529712000000,

"key\_as\_string" : "2018-06-23 00:00:00",

"doc\_count" : 3,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "上海",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

}, {

"key" : "上",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

} ]

}

}, {

"key" : 1529625600000,

"key\_as\_string" : "2018-06-22 00:00:00",

"doc\_count" : 4,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "北京",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

}, {

"key" : "海南",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

}, {

"key" : "西安",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

} ]

}

}, {

"key" : 1529539200000,

"key\_as\_string" : "2018-06-21 00:00:00",

"doc\_count" : 2,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "北京",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

} ]

}

} ]

}

}

}

数据二的请求与结果：

查询请求：

{

"size" : 20,

"query" : {

"bool" : {

"must" : {

"range" : {

"queryDate" : {

"from" : "2018-06-21",

"to" : "2018-06-24",

"format" : "yyyy-MM-dd",

"include\_lower" : false,

"include\_upper" : false

}

}

}

}

},

"aggregations" : {

"queryDate\_count" : {

"terms" : {

"field" : "queryDate",

"order" : {

"\_term" : "asc"

}

},

"aggregations" : {

"depCity\_count " : {

"terms" : {

"field" : "depCity",

"order" : {

"depCity\_count" : "desc"

}

},

"aggregations" : {

"depCity\_count" : {

"value\_count" : {

"field" : "depCity"

}

}

}

}

}

}

}

}

2018-06-14 09:39:01,628 main INFO [com.jd.es.example.Main:main] - 查询结果：

{

"took" : 1,

"timed\_out" : false,

"\_shards" : {

"total" : 4,

"successful" : 4,

"failed" : 0

},

"hits" : {

"total" : 7,

"max\_score" : 1.0,

"hits" : [ {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "6",

"\_score" : 1.0,

"\_source":{"depCity":"上","arrCity":"北京","queryDate":"2018-06-23 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "7",

"\_score" : 1.0,

"\_source":{"depCity":"上海","arrCity":"北京","queryDate":"2018-06-23 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "8",

"\_score" : 1.0,

"\_source":{"depCity":"上海","queryDate":"2018-06-23 23:00:00","userPin":"huyanxia5"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "9",

"\_score" : 1.0,

"\_source":{"depCity":"北京","arrCity":"上海","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "1",

"\_score" : 1.0,

"\_source":{"depCity":"北京","arrCity":"上海","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "4",

"\_score" : 1.0,

"\_source":{"depCity":"西安","arrCity":"延安","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "5",

"\_score" : 1.0,

"\_source":{"depCity":"海南","arrCity":"山东","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

} ]

},

"aggregations" : {

"queryDate\_count" : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : 1529625600000,

"key\_as\_string" : "2018-06-22 00:00:00",

"doc\_count" : 4,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "北京",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

}, {

"key" : "海南",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

}, {

"key" : "西安",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

} ]

}

}, {

"key" : 1529712000000,

"key\_as\_string" : "2018-06-23 00:00:00",

"doc\_count" : 2,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "上",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

}, {

"key" : "上海",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

} ]

}

}, {

"key" : 1529794800000,

"key\_as\_string" : "2018-06-23 23:00:00",

"doc\_count" : 1,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "上海",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

} ]

}

} ]

}

}

}

### 时间以天分组

Sql:

Select count(\*) as depCity\_count from table where queryDate>="2018-06-21 00:00:00" and queryDate<"2018-06-24 00:00:00" group by queryDate的天,depCity order by queryDate asc,depCity\_count desc

//时间范围查询  
BoolQueryBuilder boolQueryBuilder = QueryBuilders.*boolQuery*();  
boolQueryBuilder.must(QueryBuilders.*rangeQuery*("queryDate").format("yyyy-MM-dd").gte("2018-06-21").lt("2018-06-24"));  
  
//group by depCity order by count(\*) desc  
TermsBuilder teamAgg= AggregationBuilders.*terms*("depCity\_count ").field("depCity").subAggregation(AggregationBuilders.*count*("depCity\_count").field("depCity")).order(Terms.Order.*aggregation*("depCity\_count", false));  
  
//日期以天分组升序，与teamAgg作为子聚合  
DateHistogramBuilder dateAgg= AggregationBuilders.*dateHistogram*("queryDate\_count")  
 .field("queryDate")  
 .interval(DateHistogramInterval.*DAY*).order(Histogram.Order.*KEY\_ASC*).subAggregation(teamAgg);  
//先根据日期范围查询，再group by queryDate（具体到秒）并升序，再group by depCity order by count(\*) desc  
SearchRequestBuilder srb = Tool.*CLIENT*.prepareSearch(Tool.*INDEX\_NAME*).setTypes(Tool.*TYPE\_NAME*).setQuery(boolQueryBuilder).addAggregation(dateAgg).setSize(20);//setSize(3) 可以设置查询的文档数的大小，默认10  
*logger*.info("查询请求：\n" + srb.toString());  
SearchResponse sResponse = srb.get();  
*logger*.info("查询结果：\n" + sResponse.toString());

查询请求：

{

"size" : 20,

"query" : {

"bool" : {

"must" : {

"range" : {

"queryDate" : {

"from" : "2018-06-21",

"to" : "2018-06-24",

"format" : "yyyy-MM-dd",

"include\_lower" : true,

"include\_upper" : false

}

}

}

}

},

"aggregations" : {

"queryDate\_count" : {

"date\_histogram" : {

"field" : "queryDate",

"interval" : "1d",

"order" : {

"\_key" : "asc"

}

},

"aggregations" : {

"depCity\_count " : {

"terms" : {

"field" : "depCity",

"order" : {

"depCity\_count" : "desc"

}

},

"aggregations" : {

"depCity\_count" : {

"value\_count" : {

"field" : "depCity"

}

}

}

}

}

}

}

}

2018-06-14 10:16:00,047 main INFO [com.jd.es.example.Main:main] - 查询结果：

{

"took" : 2,

"timed\_out" : false,

"\_shards" : {

"total" : 4,

"successful" : 4,

"failed" : 0

},

"hits" : {

"total" : 9,

"max\_score" : 1.0,

"hits" : [ {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "6",

"\_score" : 1.0,

"\_source":{"depCity":"上","arrCity":"北京","queryDate":"2018-06-23 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "7",

"\_score" : 1.0,

"\_source":{"depCity":"上海","arrCity":"北京","queryDate":"2018-06-23 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "8",

"\_score" : 1.0,

"\_source":{"depCity":"上海","queryDate":"2018-06-23 23:00:00","userPin":"huyanxia5"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "9",

"\_score" : 1.0,

"\_source":{"depCity":"北京","arrCity":"上海","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "1",

"\_score" : 1.0,

"\_source":{"depCity":"北京","arrCity":"上海","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "2",

"\_score" : 1.0,

"\_source":{"depCity":"北京","arrCity":"三亚","queryDate":"2018-06-21 00:00:00","userPin":"huyanxia1"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "4",

"\_score" : 1.0,

"\_source":{"depCity":"西安","arrCity":"延安","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "5",

"\_score" : 1.0,

"\_source":{"depCity":"海南","arrCity":"山东","queryDate":"2018-06-22 00:00:00","userPin":"huyanxia"}

}, {

"\_index" : "air\_flight\_query\_index",

"\_type" : "air\_flight\_query\_type",

"\_id" : "10",

"\_score" : 1.0,

"\_source":{"depCity":"北京","arrCity":"三亚","queryDate":"2018-06-21 00:00:00","userPin":"huyanxia1"}

} ]

},

"aggregations" : {

"queryDate\_count" : {

"buckets" : [ {

"key\_as\_string" : "2018-06-21 00:00:00",

"key" : 1529539200000,

"doc\_count" : 2,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "北京",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

} ]

}

}, {

"key\_as\_string" : "2018-06-22 00:00:00",

"key" : 1529625600000,

"doc\_count" : 4,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "北京",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

}, {

"key" : "海南",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

}, {

"key" : "西安",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

} ]

}

}, {

"key\_as\_string" : "2018-06-23 00:00:00",

"key" : 1529712000000,

"doc\_count" : 3,

"depCity\_count " : {

"doc\_count\_error\_upper\_bound" : 0,

"sum\_other\_doc\_count" : 0,

"buckets" : [ {

"key" : "上海",

"doc\_count" : 2,

"depCity\_count" : {

"value" : 2

}

}, {

"key" : "上",

"doc\_count" : 1,

"depCity\_count" : {

"value" : 1

}

} ]

}

} ]

}

}

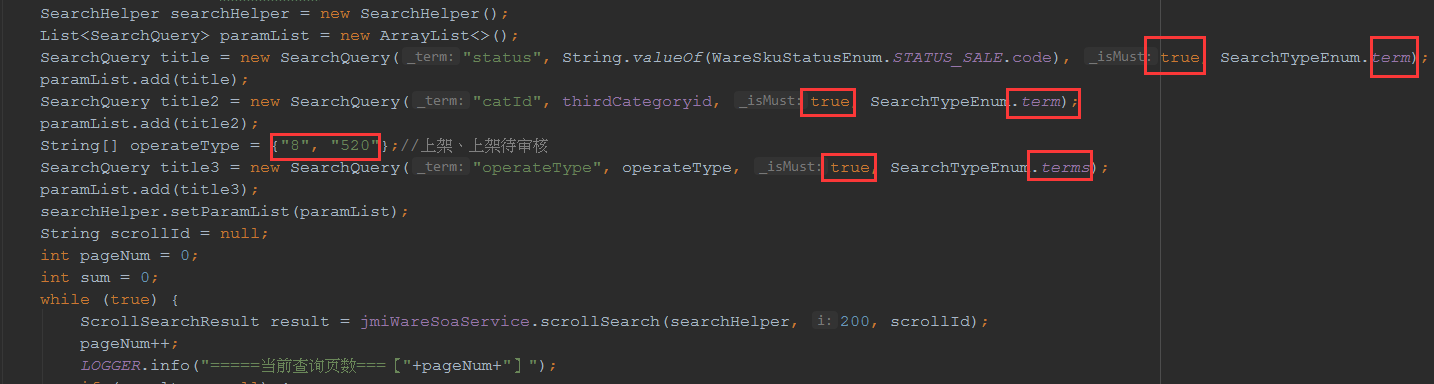
}

### 分页查询

[Elasticsearch——分页查询From&Size VS scroll](https://blog.csdn.net/z69183787/article/details/70139894)

From&Size是深翻，scroll是快照

[ElasticSearch - 解决ES的深分页问题 (游标 scroll)](https://www.jianshu.com/p/f4d322415d29)



# 天眼

查询无结果的：

{"data":{"userQueryLine":{"arriCode":null,"arriName":"上海","deptCode":null,"deptName":"北京","isHaveResult":1,"queryDate":null,"queryEndDate":"2019-03-21","queryNum":null,"queryStartDate":"2019-03-21"},"pagerData":{"data":[],"pageNo":1,"pageSize":15,"totalPage":0,"totalRecord":0}},"errorMessage":null,"status":0}

查询有结果的：

{"data":{"userQueryLine":{"arriCode":null,"arriName":"上海","deptCode":null,"deptName":"北京","isHaveResult":1,"queryDate":null,"queryEndDate":"2019-03-19","queryNum":null,"queryStartDate":"2019-03-19"},"pagerData":{"data":[{"arriCode":"SHA","arriName":"上海","deptCode":"PEK","deptName":"北京","isHaveResult":1,"queryDate":"2019-03-19","queryEndDate":null,"queryNum":268,"queryStartDate":null}],"pageNo":1,"pageSize":15,"totalPage":1,"totalRecord":1}},"errorMessage":null,"status":0}

## 索引：

### jipiao\_user\_flight\_query\_line ：

{

"cluster\_name": "jiesi-flightskyeye",

"metadata": {

"indices": {

"jipiao\_user\_flight\_query\_line": {

"settings": {

"index": {

"number\_of\_shards": "4",

"max\_result\_window": "500000",

"creation\_date": "1534915094065",

"number\_of\_replicas": "2",

"uuid": "8oXRX317SUWcYuVhSXsMRw",

"version": {

"created": "2010299"

}

}

},

"mappings": {

"user\_flight\_query\_line": {

"\_all": {

"enabled": false

},

"properties": {

"deptName": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"queryDate": {

"format": "yyyy-MM-dd",

"store": true,

"type": "date"

},

"queryNum": {

"store": true,

"type": "long"

},

"arriCode": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"arriName": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"deptCode": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"isHaveResult": {

"index": "not\_analyzed",

"store": true,

"type": "string"

}

}

}

},

"aliases": [],

"state": "open"

}

},

"cluster\_uuid": "\_na\_",

"templates": {}

}

}

### jipiao\_user\_flight\_query\_line\_details

{

"cluster\_name": "jiesi-flightskyeye",

"metadata": {

"indices": {

"jipiao\_user\_flight\_query\_line\_details": {

"settings": {

"index": {

"number\_of\_shards": "8",

"max\_result\_window": "600000",

"creation\_date": "1534920736494",

"number\_of\_replicas": "2",

"uuid": "8sZeCEVgTCKRPbXcSVUybg",

"version": {

"created": "2010299"

}

}

},

"mappings": {

"user\_flight\_query\_line\_details": {

"\_all": {

"enabled": false

},

"properties": {

"deptName": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"displayVenderIds": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"extendTxt": {

"type": "string"

},

"doingVenderIds": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"uuid": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"depDate": {

"type": "string"

},

"isHaveResult": {

"store": true,

"type": "integer"

},

"userPin": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"sourceType": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"queryTime": {

"store": true,

"type": "long"

},

"doneVenderIds": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"arriCode": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"arriName": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"deptCode": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"deptDate": {

"format": "yyyy-MM-dd",

"store": true,

"type": "date"

}

}

}

},

"aliases": [],

"state": "open"

}

},

"cluster\_uuid": "\_na\_",

"templates": {}

}

}

### jipiao\_user\_flight\_query\_log

{

"settings": {

"index": {

"number\_of\_shards": "4",

"number\_of\_replicas": "1"

}

},

"mappings": {

"user\_flight\_query\_log": {

"\_ttl": {

"enabled": false

},

"dynamic": false,

"\_all": {

"enabled": false

},

"properties": {

"depCityCode": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"arrCityCode": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"depDate": {

"format": "yyyy-MM-dd",

"store": true,

"type": "date"

},

"queryDateTime": {

"format": "yyyy-MM-dd HH:mm:ss",

"store": true,

"type": "date"

},

"userPin": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"uuid": {

"index": "not\_analyzed",

"store": true,

"type": "string"

},

"request": {

"index": "not\_analyzed",

"type": "string"

},

"response": {  
           "index": "no",  
           "type": "string"  
         }

}

}

}

}